

MPFE

Conrad Burden, Sylvain Forêt, Peijie Lin

May 30, 2026

Many of the known mechanisms driving gene regulation fall into the category of epigenomic modifications. DNA methylation is a common epigenomic modification, in which a cytosine (C) in the genomic DNA sequence can be altered by the addition of a methyl group. Methylation patterns can be detected by treating DNA with bisulphite, which converts unmethylated cytosines to uracils while leaving methylated cytosines intact. This can be carried out at the whole genome level (whole-genome bisulfite sequencing) or at specific loci (PCR amplicons, capture or reduced representation bisulfite sequencing). The resulting reads can be mapped to a reference and methylation patterns inferred.

However, the bisulphite conversion is not 100% efficient, and this introduces errors in the observed distribution of methylation patterns. A second source of errors is the sequencing error. **MPFE** (for **M**ethylation **P**atterns **F**requency **E**stimation) [1] calculates the estimated distribution over methylation patterns based on an input of methylation pattern count data, an incomplete conversion rate and site-dependent read error rates.

The main component of the package is the function `estimatePatterns()`, which generates a table of estimates $\hat{\theta}_i$ of the distribution over methylation patterns, and a list of patterns identified as spurious. Input to the function is a data frame listing the methylation patterns in the first column followed by a number of columns of count data (one column per sample). Estimation will be performed on all columns by default unless specified by the variable `column`. The non-conversion and sequencing error rates are specified by the parameters `epsilon` and `eta` respectively. The parameter `eta` can be specified globally or as a site-dependent array with length equal to the number of CpG sites in sequence of interest. The boolean variable `fast` enables either a fast implementation (default) which ignores those patterns for which the observed read count is zero or a slow implementation. The parameters `steps` and `reltol` are passed to the function `constrOptim()` to control the accuracy of the determination of the maximum log-likelihood.

A second function in the package is `plotPatterns()`. Input to this function is a data frame, obtained from the output of `estimatePatterns()`. The output of `plotPatterns()` is a plot that compares the observed read distribution with the estimated distribution. The parameters `yLimit1` and `yLimit2` control the range of the y-axis on the plots produced.

In the following example, the input is the table of counts `patternsExample`. We analyse the second column. The parameter `epsilon` is 0.01, while the parameter `eta` is not specified and by default is 0.

```
> library(MPFE)
> data(patternsExample)
> patternsExample
```

| | mPattern | k1 | k2 |
|----|----------|-----|------|
| 1 | m00000 | 629 | 2257 |
| 2 | m00001 | 26 | 90 |
| 3 | m00010 | 20 | 75 |
| 4 | m00011 | 2 | 3 |
| 5 | m00100 | 24 | 82 |
| 6 | m00101 | 3 | 0 |
| 7 | m00110 | 1 | 11 |
| 8 | m00111 | 0 | 0 |
| 9 | m01000 | 23 | 80 |
| 10 | m01001 | 0 | 0 |
| 11 | m01010 | 1 | 1 |
| 12 | m01011 | 0 | 0 |
| 13 | m01100 | 1 | 5 |
| 14 | m01110 | 0 | 0 |
| 15 | m10000 | 28 | 69 |
| 16 | m10001 | 1 | 2 |
| 17 | m10010 | 0 | 2 |
| 18 | m10011 | 0 | 0 |
| 19 | m10100 | 0 | 7 |
| 20 | m11000 | 3 | 1 |
| 21 | m11001 | 0 | 0 |

```
> estimatePatterns(patternsExample, epsilon=0.01, column=2)
```

```
[[1]]
```

| | pattern | coverage | observedDistribution | estimatedDistribution | spurious |
|---|---------|----------|----------------------|-----------------------|----------|
| 1 | 00000 | 2257 | 0.8405959032 | 0.8841393438 | FALSE |
| 2 | 00001 | 90 | 0.0335195531 | 0.0253622615 | FALSE |
| 3 | 00010 | 75 | 0.0279329609 | 0.0201750324 | FALSE |
| 4 | 00011 | 3 | 0.0011173184 | 0.0005734713 | FALSE |
| 5 | 00100 | 82 | 0.0305400372 | 0.0226428932 | FALSE |
| 6 | 00110 | 11 | 0.0040968343 | 0.0035800423 | FALSE |
| 7 | 01000 | 80 | 0.0297951583 | 0.0217193545 | FALSE |
| 8 | 01010 | 1 | 0.0003724395 | 0.0000000000 | TRUE |

| | | | | | |
|----|-------|----|--------------|--------------|-------|
| 9 | 01100 | 5 | 0.0018621974 | 0.0013301434 | FALSE |
| 10 | 10000 | 69 | 0.0256983240 | 0.0178635791 | FALSE |
| 11 | 10001 | 2 | 0.0007448790 | 0.0002242299 | FALSE |
| 12 | 10010 | 2 | 0.0007448790 | 0.0002761306 | FALSE |
| 13 | 10100 | 7 | 0.0026070764 | 0.0021135181 | FALSE |
| 14 | 11000 | 1 | 0.0003724395 | 0.0000000000 | TRUE |

Note that in this example two patterns have been identified as spurious; they are patterns 01010 and 11000.

The following example uses the same input table. The `column` variable is not specified, so the function `estimatePatterns()` by default applies to both columns of counts. The sequencing error rate `eta` is specified as a site-dependent array.

```
> estimates <- estimatePatterns(patternsExample,
+                               epsilon=0.01,
+                               eta=c(0.008, 0.01, 0.01, 0.01, 0.008))
> estimates
```

```
[[1]]
```

| | pattern | coverage | observedDistribution | estimatedDistribution | spurious |
|----|---------|----------|----------------------|-----------------------|----------|
| 1 | 00000 | 629 | 0.825459318 | 0.9088748331 | FALSE |
| 2 | 00001 | 26 | 0.034120735 | 0.0200854806 | FALSE |
| 3 | 00010 | 20 | 0.026246719 | 0.0099129217 | FALSE |
| 4 | 00011 | 2 | 0.002624672 | 0.0017646709 | FALSE |
| 5 | 00100 | 24 | 0.031496063 | 0.0155305960 | FALSE |
| 6 | 00101 | 3 | 0.003937008 | 0.0030002140 | FALSE |
| 7 | 00110 | 1 | 0.001312336 | 0.0004654103 | FALSE |
| 8 | 01000 | 23 | 0.030183727 | 0.0141238683 | FALSE |
| 9 | 01010 | 1 | 0.001312336 | 0.0004935455 | FALSE |
| 10 | 01100 | 1 | 0.001312336 | 0.0003809915 | FALSE |
| 11 | 10000 | 28 | 0.036745407 | 0.0221025625 | FALSE |
| 12 | 10001 | 1 | 0.001312336 | 0.0002793763 | FALSE |
| 13 | 11000 | 3 | 0.003937008 | 0.0029855293 | FALSE |

```
[[2]]
```

| | pattern | coverage | observedDistribution | estimatedDistribution | spurious |
|---|---------|----------|----------------------|-----------------------|----------|
| 1 | 00000 | 2257 | 0.8405959032 | 0.9257488811 | FALSE |
| 2 | 00001 | 90 | 0.0335195531 | 0.0183955504 | FALSE |
| 3 | 00010 | 75 | 0.0279329609 | 0.0115882333 | FALSE |
| 4 | 00011 | 3 | 0.0011173184 | 0.0002272565 | FALSE |
| 5 | 00100 | 82 | 0.0305400372 | 0.0141446220 | FALSE |
| 6 | 00110 | 11 | 0.0040968343 | 0.0032943413 | FALSE |
| 7 | 01000 | 80 | 0.0297951583 | 0.0127242916 | FALSE |

```
> plotPatterns(estimates[[2]])
```

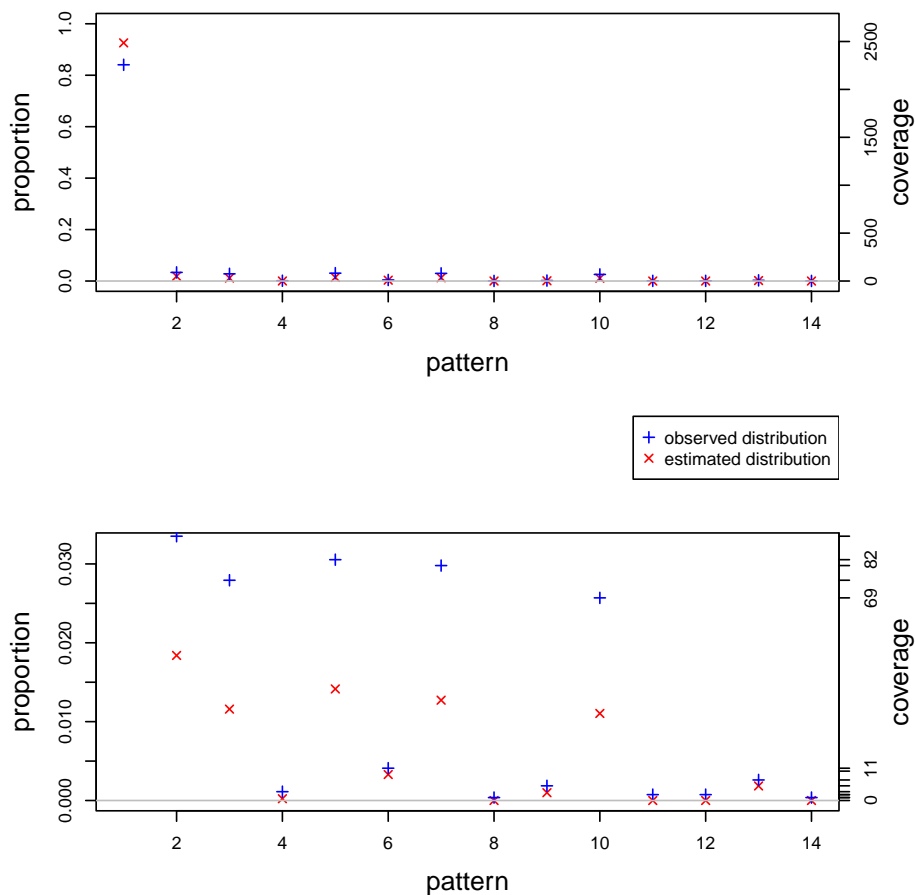


Figure 1: Plot of observed and estimated frequencies with `plotPatterns`

| | | | | | |
|----|-------|----|--------------|--------------|-------|
| 8 | 01010 | 1 | 0.0003724395 | 0.0000000000 | TRUE |
| 9 | 01100 | 5 | 0.0018621974 | 0.0009913136 | FALSE |
| 10 | 10000 | 69 | 0.0256983240 | 0.0110400877 | FALSE |
| 11 | 10001 | 2 | 0.0007448790 | 0.0000000000 | TRUE |
| 12 | 10010 | 2 | 0.0007448790 | 0.0000000000 | TRUE |
| 13 | 10100 | 7 | 0.0026070764 | 0.0018454224 | FALSE |
| 14 | 11000 | 1 | 0.0003724395 | 0.0000000000 | TRUE |

The output is a list of two data frames. We plot the observed and estimated pattern of the second data frame on figure 1. Two plots are produced: the lower plot is the expanded version of the upper plot.

We can also plot a map of the patterns and their frequencies. For instance, figure 2 illustrates different ways to plot all the patterns with frequency of 50% or less.

```

> par(mfrow=c(2, 2))
> patternMap(estimates[[1]],
+           maxFreq=0.5,
+           main='Estimated frequencies')
> patternMap(estimates[[1]],
+           estimatedDistribution=FALSE,
+           maxFreq=0.5,
+           topDown=FALSE,
+           main='Observed frequencies')
> patternMap(estimates[[1]],
+           maxFreq=0.5,
+           methCol=colorRampPalette(c('red', 'blue')),
+           unMethCol='lightgrey',
+           main='Estimated frequencies')
> patternMap(estimates[[1]],
+           estimatedDistribution=FALSE,
+           maxFreq=0.5,
+           methCol=c('bisque4', 'azure4'),
+           unMethCol=c('beige', 'azure'),
+           main='Observed frequencies')

```

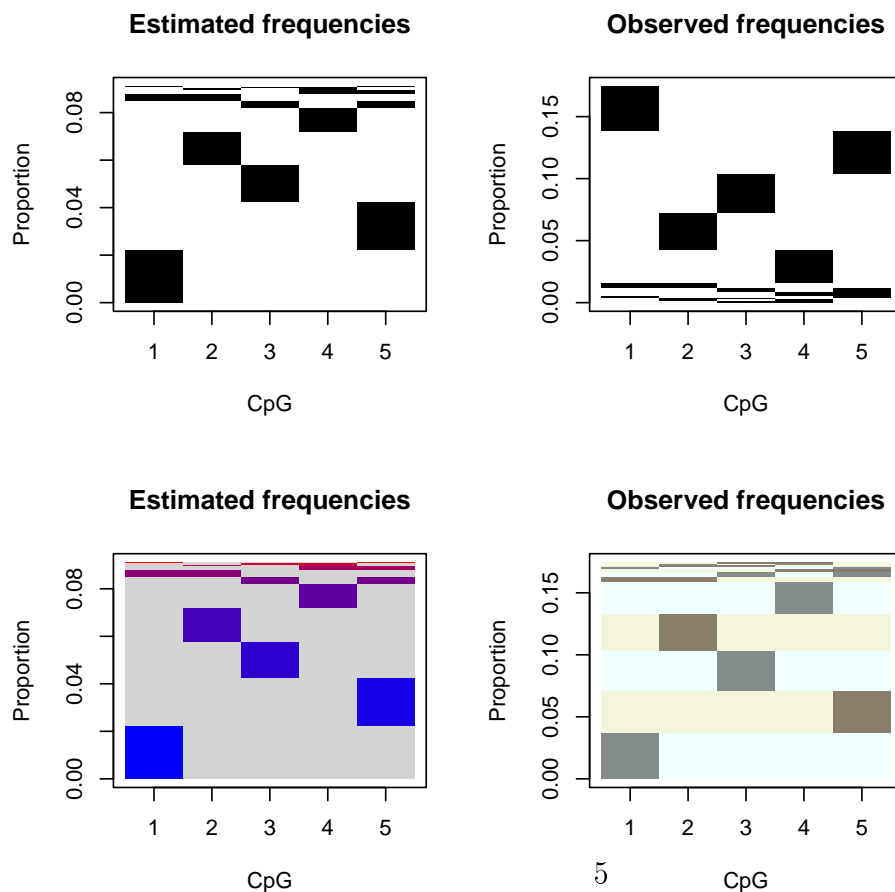


Figure 2: Different ways to plot the estimated frequencies with `patternMap`

References

- [1] Lin, P., Forêt, S., Wilson, S.R. and Burden, C.J., *Estimation of the methylation pattern distribution from deep sequencing data*. BMC Bioinformatics 2015, 16:145
doi:10.1186/s12859-015-0600-6